Computational Soundness, Co-Induction and Encryption Cycles

Daniele Micciancio

Department of Computer Science and Engineering
University of California, San Diego

June 1, 2010 (Eurocrypt'10, Nice/Monte Carlo)

Introduction

- Computational Cryptography
  - Cryptographic functions are modeled as algorithms
  - Messages exchanged by parties are bitstrings
  - Security holds against arbitrary polynomial time adversaries
- Symbolic Security (Dolev-Yao model)
  - Messages are symbolic expressions
  - Cryptographic functions work on abstract data types
  - Security holds against symbolic adversaries that respect the abstraction

Introduction

- Computational Cryptography
  - Cryptographic functions are modeled as algorithms
  - Messages exchanged by parties are bitstrings
  - Security holds against arbitrary polynomial time adversaries
- Symbolic Security (Dolev-Yao model)
  - Messages are symbolic expressions
  - Cryptographic functions work on abstract data types
  - Security holds against symbolic adversaries that respect the abstraction

Introduction

- Computational Cryptography
  - Cryptographic functions are modeled as algorithms
  - Messages exchanged by parties are bitstrings
  - Security holds against arbitrary polynomial time adversaries
- Symbolic Security (Dolev-Yao model)
  - Messages are symbolic expressions
  - Cryptographic functions work on abstract data types
  - Security holds against symbolic adversaries that respect the abstraction

Computational vs Symbolic Security

- Computational cryptography
  - Strong security guarantees √
  - Allows to define new cryptographic primitives √
  - Proofs are often complex ✗
- Symbolic Security
  - Much weaker security guarantees ✗
  - Cryptography is hard-wired into model ✗
  - Security proofs can be mechanically verified or automated √

Computational vs Symbolic Security

- Computational cryptography
  - Strong security guarantees ✓
  - Allows to define new cryptographic primitives ✓
  - Proofs are often complex ✗
- Symbolic Security
  - Much weaker security guarantees ✗
  - Cryptography is hard-wired into model ✗
  - Security proofs can be mechanically verified or automated ✓

Computational vs Symbolic Security

- Computational cryptography
  - Strong security guarantees √
  - Allows to define new cryptographic primitives √
  - Proofs are often complex ✗
- Symbolic Security
  - Much weaker security guarantees ✗
  - Cryptography is hard-wired into model ✗
  - Security proofs can be mechanically verified or automated √

Computational Soundness

- Goal: achieve the best of both words
  - Symbolic (possibly mechanized) security proofs
  - Strong security guarantees againt any polynomial time attacker
- Brief history
  - (Abadi,Rogaway 2002) First result of this kind. Limited to single message protocols and passive (eavesdropping) adversaries.
  - Many extensions since then: more cryptograhic primitives, active attacks, universal composability (Abadi, Adao, Bana, Backes, Canetti, Cortier, Jurjens, Gordon, Herzog, Laud, Mitchell, Micciancio, Panjwani, Pfitzmann, Ramanathan, Rogaway, Teague, Scedrov, Warinschi, Vene, . . .)
- This work:
  - Back to basic Abadi-Rogaway model
  - Revisit framework/approach to defining adversarial knowledge

Computational Soundness

- Goal: achieve the best of both words
  - Symbolic (possibly mechanized) security proofs
  - Strong security guarantees againt any polynomial time attacker
- Brief history
  - (Abadi,Rogaway 2002) First result of this kind. Limited to single message protocols and passive (eavesdropping) adversaries.
  - Many extensions since then: more cryptograhic primitives, active attacks, universal composability (Abadi, Adao, Bana, Backes, Canetti, Cortier, Jurjens, Gordon, Herzog, Laud, Mitchell, Micciancio, Panjwani, Pfitzmann, Ramanathan, Rogaway, Teague, Scedrov, Warinschi, Vene, . . .)
- This work:
  - Back to basic Abadi-Rogaway model
  - Revisit framework/approach to defining adversarial knowledge

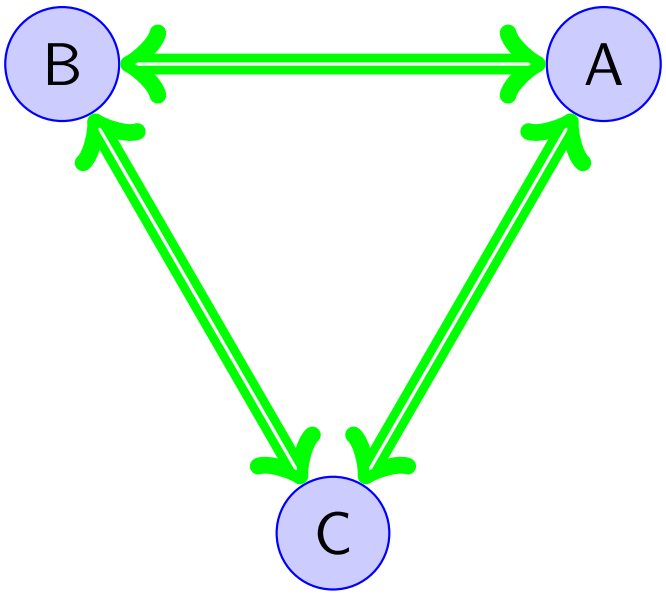
Computational Soundness

- Goal: achieve the best of both words
  - Symbolic (possibly mechanized) security proofs
  - Strong security guarantees againt any polynomial time attacker
- Brief history
  - (Abadi,Rogaway 2002) First result of this kind. Limited to single message protocols and passive (eavesdropping) adversaries.
  - Many extensions since then: more cryptograhic primitives, active attacks, universal composability (Abadi, Adao, Bana, Backes, Canetti, Cortier, Jurjens, Gordon, Herzog, Laud, Mitchell, Micciancio, Panjwani, Pfitzmann, Ramanathan, Rogaway, Teague, Scedrov, Warinschi, Vene, . . .)
- This work:
  - Back to basic Abadi-Rogaway model
  - Revisit framework/approach to defining adversarial knowledge

Outline

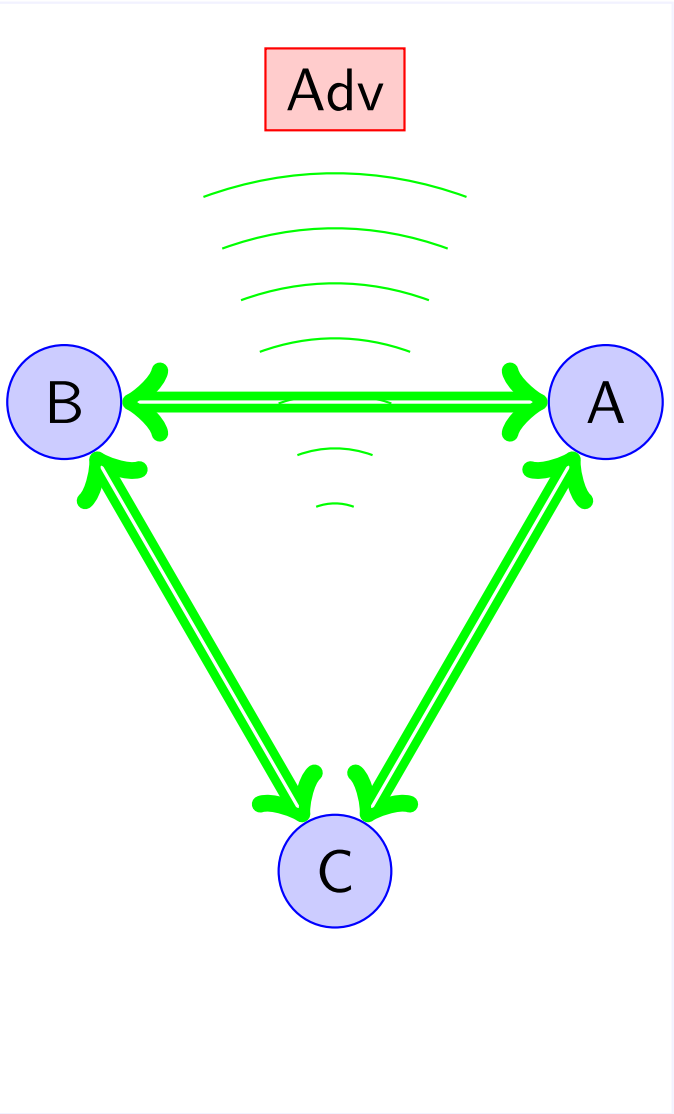
Outline

The Abadi-Rogaway model



- Protocol parties
  - are given some initial knowledge (e.g., public keys, own secret key, etc.)
  - interact by exchanging messages
- Adversary
  - Passively eavesdrop communications between parties
  - Infers knowledge from messages

The Abadi-Rogaway model



- Protocol parties
  - are given some initial knowledge (e.g., public keys, own secret key, etc.)
  - interact by exchanging messages
- Adversary
  - Passively eavesdrop communications between parties
  - Infers knowledge from messages

Some questions

- What kind of messages are exchanged among the parties?
- How is the adversarial knowledge represented?
- What does the adversary know after seeing a sequence of messages?

Messages

- Keys: K_1, K_2, ...
- Data: D_1, D_1, ...
- Messages: $E ::= K \mid D \mid (E_0, E_1) \mid \{E\}_K$
- Example: Hybrid encryption
  - K_1: long term key, K_2: session key
  - $(\{K_2\}_{K_1}, \{D\}_{K_2})$.
- Extensions:
  - Pseudorandom keys: $K ::= G_0(K); G_1(K)$
  - Secret sharing: $K ::= S_1(K), \ldots, S_n(K)$
  - Hashing: $E ::= h(E)$, etc.

Messages

- Keys: K_1, K_2, ...
- Data: D_1, D_1, ...
- Messages: $E ::= K \mid D \mid (E_0, E_1) \mid \{E\}_K$
- Example: Hybrid encryption
  - K_1: long term key, K_2: session key
  - $(\{K_2\}_{K_1}, \{D\}_{K_2})$.
- Extensions:
  - Pseudorandom keys: $K ::= G_0(K); G_1(K)$
  - Secret sharing: $K ::= S_1(K), \ldots, S_n(K)$
  - Hashing: $E ::= h(E)$, etc.

Messages

- Keys: K_1, K_2, ...
- Data: D_1, D_1, ...
- Messages: $E ::= K \mid D \mid (E_0, E_1) \mid \{E\}_K$
- Example: Hybrid encryption
  - K_1: long term key, K_2: session key
  - $(\{K_2\}_{K_1}, \{D\}_{K_2})$.
- Extensions:
  - Pseudorandom keys: $K ::= G_0(K); G_1(K)$
  - Secret sharing: $K ::= S_1(K), \ldots, S_n(K)$
  - Hashing: $E ::= h(E)$, etc.

Adversarial knowledge

- Keys can be either completely secret or known to the adversary
- Adversarial knowledge is represented by a set of known keys S
- An adversary knowing keys S, when intercepts a message E, "sees" a pattern

$$
\begin{aligned}
\mathbf{pat}(K, S) &= K \\
\mathbf{pat}(D, S) &= D \\
\mathbf{pat}((E_1, E_2), S) &= (\mathbf{pat}(E_1, S), \mathbf{pat}(E_2, S)) \\
\mathbf{pat}(\{E\}_K, S) &= \begin{cases} \{\mathbf{pat}(E, S)\}_K & \text{if } K \in S \\ \Box & \text{otherwise} \end{cases}
\end{aligned}
$$

Adversarial knowledge

- Keys can be either completely secret or known to the adversary
- Adversarial knowledge is represented by a set of known keys S
- An adversary knowing keys S, when intercepts a message E, "sees" a pattern

$$
\begin{aligned}
\mathbf{pat}(K, S) &= K \\
\mathbf{pat}(D, S) &= D \\
\mathbf{pat}((E_1, E_2), S) &= (\mathbf{pat}(E_1, S), \mathbf{pat}(E_2, S)) \\
\mathbf{pat}(\{E\}_K, S) &= \begin{cases} \{\mathbf{pat}(E, S)\}_K & \text{if } K \in S \\ \Box & \text{otherwise} \end{cases}
\end{aligned}
$$

Computational Semantics

- Expressions are evaluated using cryptographic algorithms
- The result of an expression E is a probability distribution over bitstrings $[\![E]\!]$
- If two expressions are computationally indistinguishable $[\![E_1]\!] \approx [\![E_2]\!]$, they reveal the same amount of information.

Computationally Sound Symbolic Semantics

- Symbolic semantics map expressions (messages) E to patterns
- Two expressions are symbolically equivalent if they map to the same pattern
- The symbolic semantics is computationally sound if whenever E_1 and E_2 are symbolically equivalent, it holds that $[\![E_1]\!] \approx [\![E_2]\!]$.

Outline

A simple example

- The adversary intercepts the messages $E = (K_1, \{K_2\}_{K_3}, \{(\{D_1\}_{K_2}, D_2, \}_{K_1})$
- The adversary learns K_1 because it is sent in clear, but not K_2 or K_3. The adversarial knowledge is $S = \{K_1\}$.
- The adversary's "view" of the messages is $\mathbf{pat}(E, \{K_1\}) = (K_1, \Box, \{(\Box, D_2, \}_{K_1})$

A simple example

- The adversary intercepts the messages $E = (K_1, \{K_2\}_{K_3}, \{(\{D_1\}_{K_2}, D_2, \}_{K_1})$
- The adversary learns K_1 because it is sent in clear, but not K_2 or K_3. The adversarial knowledge is $S = \{K_1\}$.
- The adversary's "view" of the messages is $\mathbf{pat}(E, \{K_1\}) = (K_1, \Box, \{(\Box, D_2, \}_{K_1})$

A simple example

- The adversary intercepts the messages $E = (K_1, \{K_2\}_{K_3}, \{(\{D_1\}_{K_2}, D_2, \}_{K_1})$
- The adversary learns K_1 because it is sent in clear, but not K_2 or K_3. The adversarial knowledge is $S = \{K_1\}$.
- The adversary's "view" of the messages is $\mathbf{pat}(E, \{K_1\}) = (K_1, \Box, \{(\Box, D_2, \}_{K_1})$

Defining the adversarial knowledge by induction

- Adversary sees $\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4}, K_1$
- Keys known to the adversary
  - Initial knowledge: $\emptyset$
  - K_1: sent in the clear
  - K_4: encrypted under K_1
  - K_5: encrypted under K_4
  - K_3: double encrypted under K_1 and K_4
- Adversarial knowledge: $S = \{K_1, K_3, K_4, K_5\}$

Defining the adversarial knowledge by induction

- Adversary sees $\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4}, K_1$
- Keys known to the adversary
  - Initial knowledge: $\emptyset$
  - K_1: sent in the clear
  - K_4: encrypted under K_1
  - K_5: encrypted under K_4
  - K_3: double encrypted under K_1 and K_4
- Adversarial knowledge: $S = \{K_1, K_3, K_4, K_5\}$

Defining the adversarial knowledge by induction

- Adversary sees $\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4}, K_1$
- Keys known to the adversary
  - Initial knowledge: $\emptyset$
  - K_1: sent in the clear
  - K_4: encrypted under K_1
  - K_5: encrypted under K_4
  - K_3: double encrypted under K_1 and K_4
- Adversarial knowledge: $S = \{K_1, K_3, K_4, K_5\}$

Defining the adversarial knowledge by induction

- Adversary sees $\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4}, K_1$
- Keys known to the adversary
  - Initial knowledge: $\emptyset$
  - K_1: sent in the clear
  - K_4: encrypted under K_1
  - K_5: encrypted under K_4
  - K_3: double encrypted under K_1 and K_4
- Adversarial knowledge: $S = \{K_1, K_3, K_4, K_5\}$

Defining the adversarial knowledge by induction

- Adversary sees $\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4}, K_1$
- Keys known to the adversary
  - Initial knowledge: $\emptyset$
  - K_1: sent in the clear
  - K_4: encrypted under K_1
  - K_5: encrypted under K_4
  - K_3: double encrypted under K_1 and K_4
- Adversarial knowledge: $S = \{K_1, K_3, K_4, K_5\}$

Defining the adversarial knowledge by induction

- Adversary sees $\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4}, K_1$
- Keys known to the adversary
  - Initial knowledge: $\emptyset$
  - K_1: sent in the clear
  - K_4: encrypted under K_1
  - K_5: encrypted under K_4
  - K_3: double encrypted under K_1 and K_4
- Adversarial knowledge: $S = \{K_1, K_3, K_4, K_5\}$

Defining the adversarial knowledge by induction

- Adversary sees $\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4}, K_1$
- Keys known to the adversary
  - Initial knowledge: $\emptyset$
  - K_1: sent in the clear
  - K_4: encrypted under K_1
  - K_5: encrypted under K_4
  - K_3: double encrypted under K_1 and K_4
- Adversarial knowledge: $S = \{K_1, K_3, K_4, K_5\}$

Adversarial knowledge as a fixpoint

- The adversarial knowedge defined by inductive key recovery process can be defined as a fixpoint of an appropriate operator
- Key recovery operator $\mathcal{F}_E : \mathcal{P}(\mathbf{Keys}) \to \mathcal{P}(\mathbf{Keys})$

$$
\begin{aligned}
\mathcal{F}_K(S) &= \{K\} \\
\mathcal{F}_D(S) &= \emptyset \\
\mathcal{F}_{(E_1,E_2)}(S) &= \mathcal{F}_{E_1}(S) \cup \mathcal{F}_{E_2}(S) \\
\mathcal{F}_{\{E\}_K}(S) &= \begin{cases} \mathcal{F}_E(S) & \text{if } K \in S \\ \emptyset & \text{otherwise} \end{cases}
\end{aligned}
$$

- Intuition: $\mathcal{F}_E(S)$ is the set of keys immediately recoverable from E given the ability to decrypt under the keys in S.

Adversarial knowledge as a fixpoint

- The set of keys S known to an adversary that intercepts message E, should satisfy $\mathcal{F}_E(S) = S$
- Some expressions have more than one fixpoint, e.g., $E = \{K\}_K$:
  - $\mathcal{F}_E(\emptyset) = \emptyset$
  - $\mathcal{F}_E(\{\mathbf{Keys}\}) = \{\mathbf{Keys}\}$.
- Inductive definition gives the least fixpoint:

$$\emptyset \subset \mathcal{F}_E(\emptyset) \subset \mathcal{F}_E^2(\emptyset) \subset \cdots \subset \mathrm{fix}(\mathcal{F}_E) = \mathcal{F}_E(\mathrm{fix}(\mathcal{F}_E))$$

- Dolev-Yao/Abadi-Rogaway: The symbolic semantics of E is $\mathbf{pat}(E, \mathrm{fix}(\mathcal{F}_E))$.

Adversarial knowledge as a fixpoint

- The set of keys S known to an adversary that intercepts message E, should satisfy $\mathcal{F}_E(S) = S$
- Some expressions have more than one fixpoint, e.g., $E = \{K\}_K$:
  - $\mathcal{F}_E(\emptyset) = \emptyset$
  - $\mathcal{F}_E(\{\mathbf{Keys}\}) = \{\mathbf{Keys}\}$.
- Inductive definition gives the least fixpoint:

$$\emptyset \subset \mathcal{F}_E(\emptyset) \subset \mathcal{F}_E^2(\emptyset) \subset \cdots \subset \mathrm{fix}(\mathcal{F}_E) = \mathcal{F}_E(\mathrm{fix}(\mathcal{F}_E))$$

- Dolev-Yao/Abadi-Rogaway: The symbolic semantics of E is $\mathbf{pat}(E, \mathrm{fix}(\mathcal{F}_E))$.

Adversarial knowledge as a fixpoint

- The set of keys S known to an adversary that intercepts message E, should satisfy $\mathcal{F}_E(S) = S$
- Some expressions have more than one fixpoint, e.g., $E = \{K\}_K$:
  - $\mathcal{F}_E(\emptyset) = \emptyset$
  - $\mathcal{F}_E(\{\mathbf{Keys}\}) = \{\mathbf{Keys}\}$.
- Inductive definition gives the least fixpoint:

$$\emptyset \subset \mathcal{F}_E(\emptyset) \subset \mathcal{F}_E^2(\emptyset) \subset \cdots \subset \mathrm{fix}(\mathcal{F}_E) = \mathcal{F}_E(\mathrm{fix}(\mathcal{F}_E))$$

- Dolev-Yao/Abadi-Rogaway: The symbolic semantics of E is $\mathbf{pat}(E, \mathrm{fix}(\mathcal{F}_E))$.

Adversarial knowledge as a fixpoint

- The set of keys S known to an adversary that intercepts message E, should satisfy $\mathcal{F}_E(S) = S$
- Some expressions have more than one fixpoint, e.g., $E = \{K\}_K$:
  - $\mathcal{F}_E(\emptyset) = \emptyset$
  - $\mathcal{F}_E(\{\mathbf{Keys}\}) = \{\mathbf{Keys}\}$.
- Inductive definition gives the least fixpoint:

$$\emptyset \subset \mathcal{F}_E(\emptyset) \subset \mathcal{F}_E^2(\emptyset) \subset \cdots \subset \mathrm{fix}(\mathcal{F}_E) = \mathcal{F}_E(\mathrm{fix}(\mathcal{F}_E))$$

- Dolev-Yao/Abadi-Rogaway: The symbolic semantics of E is $\mathbf{pat}(E, \mathrm{fix}(\mathcal{F}_E))$.

Computational Soundness and Encryption cycles

- Is the Abadi-Rogaway symbolic semantic computationally sound? In general, no!
- Abadi-Rogaway: if two expressions with no encryption cycles are symbolically equivalent, then they are computationally indistinguishable
- Examples of cyclic expressions:
  - $\{K\}_K$
  - $\{K_1\}_{K_2}, \{K_2\}_{K_3}, \{K_3\}_{K_1}$
- What if the expressions contain cycles?
  - Long standing open problem: come up with encryption scheme that breaks down in the precence of encryption cycles
  - Much recent work on "key dependent message" security (Boneh,Halevi,Hamburg,Ostrovsky 2008) (Applebaum,Cash,Peikert,Sahai 2009)

Computational Soundness and Encryption cycles

- Is the Abadi-Rogaway symbolic semantic computationally sound? In general, no!
- Abadi-Rogaway: if two expressions with no encryption cycles are symbolically equivalent, then they are computationally indistinguishable
- Examples of cyclic expressions:
  - $\{K\}_K$
  - $\{K_1\}_{K_2}, \{K_2\}_{K_3}, \{K_3\}_{K_1}$
- What if the expressions contain cycles?
  - Long standing open problem: come up with encryption scheme that breaks down in the precence of encryption cycles
  - Much recent work on "key dependent message" security (Boneh,Halevi,Hamburg,Ostrovsky 2008) (Applebaum,Cash,Peikert,Sahai 2009)

Computational Soundness and Encryption cycles

- Is the Abadi-Rogaway symbolic semantic computationally sound? In general, no!
- Abadi-Rogaway: if two expressions with no encryption cycles are symbolically equivalent, then they are computationally indistinguishable
- Examples of cyclic expressions:
  - $\{K\}_K$
  - $\{K_1\}_{K_2}, \{K_2\}_{K_3}, \{K_3\}_{K_1}$
- What if the expressions contain cycles?
  - Long standing open problem: come up with encryption scheme that breaks down in the precence of encryption cycles
  - Much recent work on "key dependent message" security (Boneh,Halevi,Hamburg,Ostrovsky 2008) (Applebaum,Cash,Peikert,Sahai 2009)

Computational Soundness and Encryption cycles

- Is the Abadi-Rogaway symbolic semantic computationally sound? In general, no!
- Abadi-Rogaway: if two expressions with no encryption cycles are symbolically equivalent, then they are computationally indistinguishable
- Examples of cyclic expressions:
  - $\{K\}_K$
  - $\{K_1\}_{K_2}, \{K_2\}_{K_3}, \{K_3\}_{K_1}$
- What if the expressions contain cycles?
  - Long standing open problem: come up with encryption scheme that breaks down in the precence of encryption cycles
  - Much recent work on "key dependent message" security (Boneh,Halevi,Hamburg,Ostrovsky 2008) (Applebaum,Cash,Peikert,Sahai 2009)

Computational Soundness and Encryption cycles

- Is the Abadi-Rogaway symbolic semantic computationally sound? In general, no!
- Abadi-Rogaway: if two expressions with no encryption cycles are symbolically equivalent, then they are computationally indistinguishable
- Examples of cyclic expressions:
  - $\{K\}_K$
  - $\{K_1\}_{K_2}, \{K_2\}_{K_3}, \{K_3\}_{K_1}$
- What if the expressions contain cycles?
  - Long standing open problem: come up with encryption scheme that breaks down in the precence of encryption cycles
  - Much recent work on "key dependent message" security (Boneh,Halevi,Hamburg,Ostrovsky 2008) (Applebaum,Cash,Peikert,Sahai 2009)

Our work

- Goal: better understand relation between symbolc semantics and standard computational security definition
- Technique: define adversarial knowledge as the greatest fixpoint (FIX) of $\mathcal{F}_E$ (by "co-induction")
- Intuition: assume no key is guaranteed to be secret, and prove that more and more keys are hidden to the adversary
- Results:
  - Theorem 1: The GFP semantics is computationally sound, i.e., for any two expressions E_1, E_2:
    $\mathbf{pat}(E_1, \mathrm{FIX}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathrm{FIX}(\mathcal{F}_{E_2})) \Longrightarrow [\![E_1]\!] \approx [\![E_2]\!]$
  - Theorem 2: If E is acyclic, then $\mathrm{fix}(\mathcal{F}_E) = \mathrm{FIX}(\mathcal{F}_E)$
  - Corollary: if E_1, E_2 have no encryption cycles and $\mathbf{pat}(E_1, \mathrm{fix}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathrm{fix}(\mathcal{F}_{E_2}))$, then $[\![E_1]\!] \approx [\![E_2]\!]$

Our work

- Goal: better understand relation between symbolc semantics and standard computational security definition
- Technique: define adversarial knowledge as the greatest fixpoint (FIX) of $\mathcal{F}_E$ (by "co-induction")
- Intuition: assume no key is guaranteed to be secret, and prove that more and more keys are hidden to the adversary
- Results:
  - Theorem 1: The GFP semantics is computationally sound, i.e., for any two expressions E_1, E_2:
    $\mathbf{pat}(E_1, \mathrm{FIX}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathrm{FIX}(\mathcal{F}_{E_2})) \Longrightarrow [\![E_1]\!] \approx [\![E_2]\!]$
  - Theorem 2: If E is acyclic, then $\mathrm{fix}(\mathcal{F}_E) = \mathrm{FIX}(\mathcal{F}_E)$
  - Corollary: if E_1, E_2 have no encryption cycles and $\mathbf{pat}(E_1, \mathrm{fix}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathrm{fix}(\mathcal{F}_{E_2}))$, then $[\![E_1]\!] \approx [\![E_2]\!]$

Our work

- Goal: better understand relation between symbolc semantics and standard computational security definition
- Technique: define adversarial knowledge as the greatest fixpoint (FIX) of $\mathcal{F}_E$ (by "co-induction")
- Intuition: assume no key is guaranteed to be secret, and prove that more and more keys are hidden to the adversary
- Results:
  - Theorem 1: The GFP semantics is computationally sound, i.e., for any two expressions E_1, E_2:
    $\mathbf{pat}(E_1, \mathrm{FIX}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathrm{FIX}(\mathcal{F}_{E_2})) \Longrightarrow [\![E_1]\!] \approx [\![E_2]\!]$
  - Theorem 2: If E is acyclic, then $\mathrm{fix}(\mathcal{F}_E) = \mathrm{FIX}(\mathcal{F}_E)$
  - Corollary: if E_1, E_2 have no encryption cycles and $\mathbf{pat}(E_1, \mathrm{fix}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathrm{fix}(\mathcal{F}_{E_2}))$, then $[\![E_1]\!] \approx [\![E_2]\!]$

Our work

- Goal: better understand relation between symbolc semantics and standard computational security definition
- Technique: define adversarial knowledge as the greatest fixpoint (FIX) of $\mathcal{F}_E$ (by "co-induction")
- Intuition: assume no key is guaranteed to be secret, and prove that more and more keys are hidden to the adversary
- Results:
  - Theorem 1: The GFP semantics is computationally sound, i.e., for any two expressions E_1, E_2:
    $\mathbf{pat}(E_1, \mathrm{FIX}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathrm{FIX}(\mathcal{F}_{E_2})) \Longrightarrow [\![E_1]\!] \approx [\![E_2]\!]$
  - Theorem 2: If E is acyclic, then $\mathrm{fix}(\mathcal{F}_E) = \mathrm{FIX}(\mathcal{F}_E)$
  - Corollary: if E_1, E_2 have no encryption cycles and $\mathbf{pat}(E_1, \mathrm{fix}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathrm{fix}(\mathcal{F}_{E_2}))$, then $[\![E_1]\!] \approx [\![E_2]\!]$

Our work

- Goal: better understand relation between symbolc semantics and standard computational security definition
- Technique: define adversarial knowledge as the greatest fixpoint (FIX) of $\mathcal{F}_E$ (by "co-induction")
- Intuition: assume no key is guaranteed to be secret, and prove that more and more keys are hidden to the adversary
- Results:
  - Theorem 1: The GFP semantics is computationally sound, i.e., for any two expressions E_1, E_2:
    $\mathbf{pat}(E_1, \mathsf{FIX}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathsf{FIX}(\mathcal{F}_{E_2})) \Longrightarrow [\![E_1]\!] \approx [\![E_2]\!]$
  - Theorem 2: If E is acyclic, then $\mathsf{fix}(\mathcal{F}_E) = \mathsf{FIX}(\mathcal{F}_E)$
  - Corollary: if E_1, E_2 have no encryption cycles and $\mathbf{pat}(E_1, \mathsf{fix}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathsf{fix}(\mathcal{F}_{E_2}))$, then $[\![E_1]\!] \approx [\![E_2]\!]$

Our work

- Goal: better understand relation between symbolc semantics and standard computational security definition
- Technique: define adversarial knowledge as the greatest fixpoint (FIX) of $\mathcal{F}_E$ (by "co-induction")
- Intuition: assume no key is guaranteed to be secret, and prove that more and more keys are hidden to the adversary
- Results:
  - Theorem 1: The GFP semantics is computationally sound, i.e., for any two expressions E_1, E_2:
    $\mathbf{pat}(E_1, \mathrm{FIX}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathrm{FIX}(\mathcal{F}_{E_2})) \Longrightarrow [\![E_1]\!] \approx [\![E_2]\!]$
  - Theorem 2: If E is acyclic, then $\mathrm{fix}(\mathcal{F}_E) = \mathrm{FIX}(\mathcal{F}_E)$
  - Corollary: if E_1, E_2 have no encryption cycles and $\mathbf{pat}(E_1, \mathrm{fix}(\mathcal{F}_{E_1})) = \mathbf{pat}(E_2, \mathrm{fix}(\mathcal{F}_{E_2}))$, then $[\![E_1]\!] \approx [\![E_2]\!]$

Greatest fixpoints

- Computing $\mathrm{FIX}(\mathcal{F}_E)$

  $$\mathbf{Keys} \supset \mathcal{F}_E(\mathbf{Keys}) \supset \mathcal{F}_E^2(\mathbf{Keys}) \supset \cdots \supset \mathrm{FIX}(\mathcal{F}_E) = \mathcal{F}_E(\mathrm{FIX}(\mathcal{F}_E))$$

- Reminder: $\mathcal{F}_E(S)$ is the set of keys immediately recoverable from E using the keys in S for decryption
- Example: $E = (\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4}, K_1)$
  - Initial set: $\mathbf{Keys} = \{K_1, K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E(\mathbf{Keys}) = \{K_1, K_2, K_3, K_4, K_5\} = \mathbf{Keys}$
  - $\mathrm{FIX}(\mathcal{F}_E) = \{K_1, K_2, K_3, K_4, K_5\} = \mathbf{Keys}$

Greatest fixpoints

- Computing $\mathrm{FIX}(\mathcal{F}_E)$

  $$\mathbf{Keys} \supset \mathcal{F}_E(\mathbf{Keys}) \supset \mathcal{F}_E^2(\mathbf{Keys}) \supset \cdots \supset \mathrm{FIX}(\mathcal{F}_E) = \mathcal{F}_E(\mathrm{FIX}(\mathcal{F}_E))$$

- Reminder: $\mathcal{F}_E(S)$ is the set of keys immediately recoverable from E using the keys in S for decryption
- Example: $E = (\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4}, K_1)$
  - Initial set: $\mathbf{Keys} = \{K_1, K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E(\mathbf{Keys}) = \{K_1, K_2, K_3, K_4, K_5\} = \mathbf{Keys}$
  - $\mathrm{FIX}(\mathcal{F}_E) = \{K_1, K_2, K_3, K_4, K_5\} = \mathbf{Keys}$

Another example

- Adversary sees $E = (\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4})$
- Adversarial knowledge
  - Initial set: $\mathbf{Keys} = \{K_1, K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E(\mathbf{Keys}) = \{K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E^2(\mathbf{Keys}) = \{K_2, K_5\}$
  - $\mathcal{F}_E^3(\mathbf{Keys}) = \{K_2\}$
  - $\mathcal{F}_E^4(\mathbf{Keys}) = \{K_2\}$
- Adversarial knowledge: $S = \{K_2\}$
- $\mathbf{pat}(E, \mathrm{FIX}(\mathcal{F}_E)) = \mathbf{pat}(E, \{K_2\}) = (\Box, \{K_2\}_{K_2}, \Box)$

Another example

- Adversary sees $E = (\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4})$
- Adversarial knowledge
  - Initial set: $\mathbf{Keys} = \{K_1, K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E(\mathbf{Keys}) = \{K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E^2(\mathbf{Keys}) = \{K_2, K_5\}$
  - $\mathcal{F}_E^3(\mathbf{Keys}) = \{K_2\}$
  - $\mathcal{F}_E^4(\mathbf{Keys}) = \{K_2\}$
- Adversarial knowledge: $S = \{K_2\}$
- $\mathbf{pat}(E, \mathrm{FIX}(\mathcal{F}_E)) = \mathbf{pat}(E, \{K_2\}) = (\Box, \{K_2\}_{K_2}, \Box)$

Another example

- Adversary sees $E = (\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4})$
- Adversarial knowledge
  - Initial set: $\mathbf{Keys} = \{K_1, K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E(\mathbf{Keys}) = \{K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E^2(\mathbf{Keys}) = \{K_2, K_5\}$
  - $\mathcal{F}_E^3(\mathbf{Keys}) = \{K_2\}$
  - $\mathcal{F}_E^4(\mathbf{Keys}) = \{K_2\}$
- Adversarial knowledge: $S = \{K_2\}$
- $\mathbf{pat}(E, \mathrm{FIX}(\mathcal{F}_E)) = \mathbf{pat}(E, \{K_2\}) = (\Box, \{K_2\}_{K_2}, \Box)$

Another example

- Adversary sees $E = (\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4})$
- Adversarial knowledge
  - Initial set: $\mathbf{Keys} = \{K_1, K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E(\mathbf{Keys}) = \{K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E^2(\mathbf{Keys}) = \{K_2, K_5\}$
  - $\mathcal{F}_E^3(\mathbf{Keys}) = \{K_2\}$
  - $\mathcal{F}_E^4(\mathbf{Keys}) = \{K_2\}$
- Adversarial knowledge: $S = \{K_2\}$
- $\mathbf{pat}(E, \mathrm{FIX}(\mathcal{F}_E)) = \mathbf{pat}(E, \{K_2\}) = (\square, \{K_2\}_{K_2}, \square)$

Another example

- Adversary sees $E = (\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4})$
- Adversarial knowledge
  - Initial set: $\mathbf{Keys} = \{K_1, K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E(\mathbf{Keys}) = \{K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E^2(\mathbf{Keys}) = \{K_2, K_5\}$
  - $\mathcal{F}_E^3(\mathbf{Keys}) = \{K_2\}$
  - $\mathcal{F}_E^4(\mathbf{Keys}) = \{K_2\}$
- Adversarial knowledge: $S = \{K_2\}$
- $\mathbf{pat}(E, \mathrm{FIX}(\mathcal{F}_E)) = \mathbf{pat}(E, \{K_2\}) = (\Box, \{K_2\}_{K_2}, \Box)$

Another example

- Adversary sees $E = (\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4})$
- Adversarial knowledge
  - Initial set: $\mathbf{Keys} = \{K_1, K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E(\mathbf{Keys}) = \{K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E^2(\mathbf{Keys}) = \{K_2, K_5\}$
  - $\mathcal{F}_E^3(\mathbf{Keys}) = \{K_2\}$
  - $\mathcal{F}_E^4(\mathbf{Keys}) = \{K_2\}$
- Adversarial knowledge: $S = \{K_2\}$
- $\mathbf{pat}(E, \mathrm{FIX}(\mathcal{F}_E)) = \mathbf{pat}(E, \{K_2\}) = (\Box, \{K_2\}_{K_2}, \Box)$

Another example

- Adversary sees $E = (\{K_4\}_{K_1}, \{K_2\}_{K_2}, \{\{K_3\}_{K_1}, K_5\}_{K_4})$
- Adversarial knowledge
  - Initial set: $\mathbf{Keys} = \{K_1, K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E(\mathbf{Keys}) = \{K_2, K_3, K_4, K_5\}$
  - $\mathcal{F}_E^2(\mathbf{Keys}) = \{K_2, K_5\}$
  - $\mathcal{F}_E^3(\mathbf{Keys}) = \{K_2\}$
  - $\mathcal{F}_E^4(\mathbf{Keys}) = \{K_2\}$
- Adversarial knowledge: $S = \{K_2\}$
- $\mathbf{pat}(E, \mathrm{FIX}(\mathcal{F}_E)) = \mathbf{pat}(E, \{K_2\}) = (\square, \{K_2\}_{K_2}, \square)$

Proof sketch of main soundness theorem

- Notice: $\mathcal{F}_E(S) = \{K \in \mathbf{pat}(E, S)\}$
- Properties of patterns:
  - $\mathbf{pat}(E, \mathbf{Keys}) = E$
  - $\mathbf{pat}(\mathbf{pat}(E, S), T) = \mathbf{pat}(E, S \cap T)$
  - $\{K \in \mathbf{pat}(E, S)\} \subseteq \{K \in E\}$
- Lemma: $[\![E]\!] \approx [\![\mathbf{pat}(E, \{K \in E\})]\!]$
- Corollary: $\forall i.[\![\mathbf{pat}(E, \mathcal{F}_E^i(\mathbf{Keys}))]\!] \approx [\![\mathbf{pat}(E, \mathcal{F}_E^{i+1}(\mathbf{Keys}))]\!]$.

$$\begin{aligned}
[\![E]\!] &= [\![\mathbf{pat}(E, \mathcal{F}_E^0(\mathbf{Keys}))]\!] \\
&\approx [\![\mathbf{pat}(E, \mathcal{F}_E^1(\mathbf{Keys}))]\!] \\
&\approx \cdots \\
&\approx [\![\mathbf{pat}(E, \mathcal{F}_E^n(\mathbf{Keys}))]\!] \\
&= [\![\mathbf{pat}(E, \mathsf{FIX}(\mathcal{F}_E))]\!]
\end{aligned}$$

Outline

Conclusion

- Using GFP rather than LFP in symbolic definition of adversarial knowledge gives better correspondence between symbolic and computational semantics
- If computational cryptography is the "right" way to analyze security protocols, then much work in symbolic security analysis needs to be revisited
- Our is just a very simple step, and much more work needs to the done
- Main open problem: extend "co-inductive" approach to active attacks